



THE UNIVERSITY *of* EDINBURGH

## Edinburgh Research Explorer

### Computation, Individuation, and the Received View on Representation

**Citation for published version:**

Sprevak, M 2010, 'Computation, Individuation, and the Received View on Representation', *Studies In History and Philosophy of Science Part A*, vol. 41, no. 3, pp. 260-270.  
<https://doi.org/10.1016/j.shpsa.2010.07.008>

**Digital Object Identifier (DOI):**

[10.1016/j.shpsa.2010.07.008](https://doi.org/10.1016/j.shpsa.2010.07.008)

**Link:**

[Link to publication record in Edinburgh Research Explorer](#)

**Document Version:**

Peer reviewed version

**Published In:**

Studies In History and Philosophy of Science Part A

**Publisher Rights Statement:**

© Mark Sprevak, 2010. Sprevak, M. (2010). Computation, individuation, and the received view on representation. *Studies in History and Philosophy of Science Part A*, 41(3), 260-270doi: 10.1016/j.shpsa.2010.07.008

**General rights**

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

**Take down policy**

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact [openaccess@ed.ac.uk](mailto:openaccess@ed.ac.uk) providing details, and we will remove access to the work immediately and investigate your claim.



# Computation, individuation, and the received view on representation

Mark Sprevak  
*University of Edinburgh*

8 August 2010

The ‘received view’ about computation is that all computations must involve representational content. Egan and Piccinini argue against the received view. In this paper, I focus on Egan’s arguments, claiming that they fall short of establishing that computations do not involve representational content. I provide positive arguments explaining why computation has to involve representational content, and how the representational content may be of any type (e.g. distal, broad, etc.). I also argue (*contra* Egan and Fodor) that there is no need for computational psychology to be individualistic. Finally, I draw out a number of consequences for computational individuation, proposing necessary conditions on computational identity and necessary and sufficient conditions on computational I/O equivalence of physical systems.

## 1 Introduction

What makes a physical process a computation? What is the difference between a computation and any other process? Under what conditions are two computations the same or different? These are among the key questions that a philosophical theory of physical computation should answer. The detailed shape of the answers is not yet clear. Yet there seem to be certain features that any reasonable theory of computational implementation should possess. What has been labelled ‘the received view’ is that computation must involve representational content.<sup>1</sup> According to this view, a necessary condition on any physical process counting as a computation is that it possess representational content (and that representational content may be of distal objects). The received view has come under two influential attacks, from Egan (1991, 1992, 1994, 1995) and Piccinini (2008). Egan

---

1. Philosophers with views as divergent as Churchland (1986); Crane (2003); Cummins (1989); Dennett (1971); Fodor (1998); Pylyshyn (1984); Searle (1992) hold that computation must involve representational content.

argues that one should understand computation in purely mathematical terms, Piccinini that one should understand computation mechanistically.<sup>2</sup> In this paper, I focus on Egan's argument. I argue that Egan's attack fails. The focus is on Egan, but there are points of contact throughout with Piccinini's argument and these will be noted in passing. A full discussion of Piccinini's sophisticated position has to wait until another occasion.

The purpose of this paper may appear overly negative: to show that Egan's attack against the received view fails. However, the argument is constructive in a number of ways. First, I argue that a distinction should be made between the concept of computation employed by mathematical computation theory and that used in the implementation of a computation by a physical system. Second, I argue that even if one wishes to take mathematical computation theory as a model for other computation talk, appeal to representational content is inescapable when attributing computations to physical systems. Third, I argue, *contra* Egan and Fodor, that there is no conceptual link between computational psychology and individualism. There is no reason why a computational psychology should be individualistic, or if it were to involve representation, why it should only involve narrow content. In other words, methodological solipsism is no part of the computational theory of mind. Fourth, I sketch positive arguments for why computation has to involve representational content.

The outline of the paper is as follows. In Section 2, I consider three arguments against the received view and show why they fail. The first argument is not endorsed by Egan but it has widespread currency in the philosophical literature, and is worth considering if only to get it out of the way. Egan's influential arguments receive a more detailed treatment. Egan's first argument is based on the interpretation of Marr's theory of vision. Egan argues that Marr's theory—which is a paradigm of computational explanation—does not posit semantic content. Hence, not all computational processes need involve representational content. Egan's second argument involves a dilemma concerning narrow content. She claims that anyone who accepts computational psychology must accept either an unpleasant commitment to narrow content, or drop the received view entirely. I argue that both Egan's ways of attacking the received view can be resisted. In Section 3, I turn to positive arguments for the received view. This section is not intended as a full-fledged defence of the received view, but it does highlight what I take to be the key intuitions that should underlie such a defence. The intention is to demonstrate that the received view is alive and kicking. The claim that computation must involve representation, as a received view, may appear apt for debunking, but in this case, the received view is simply true.

Before proceeding, a number of qualifications should be mentioned.

First, the received view is the claim is that representation is *essential* to computation:

**R** Computation essentially involves representational content

Where no restriction is placed on the *type* of representational content involved, e.g. no ban on distal objects serving as content. Egan and Piccinini accept that computations often do involve representational content, but they argue that such features are accidental

---

2. Cf. Bechtel and Richardson (1993); Craver (2007); Glennan (2002); Machamer, Darden and Craver (2000).

to a system's computational nature, and have no bearing on its computational identity. My claim is that such representational content is a necessary condition that does crucial work in determining computational identity. It is worth emphasising that even on this view, representation would still only be one condition on computational implementation: there are further conditions that a physical computation should satisfy, and additional properties that differentiate physical computations. However, representation does much of the hard work in answering the questions about individuation that motivate an account of physical computation. Consequently, it is a feature of physical computation that should be of special interest.

Second, discussion of the received view is often phrased in terms of a consequence that the view might have: that the computations involved in cognition essentially have their intentional content.<sup>3</sup> I wish to avoid phrasing the debate in terms of intentional content. The question of whether computation is committed to intentional content introduces a number of requirements that go beyond R, and one would not wish to pre-judge those issues when considering R. For example, intentional contents plausibly require the involvement of cognitive agents, but one would not wish to pre-judge whether computations can take place without involvement of cognitive agents. Intentional contents have a mode of presentation as well as an object or referent. But one would not wish to pre-judge whether the representations involved in a physical computation must have a mode of presentation as well as a referent. Intentional states play complex causal roles in our psychology. But one would not wish to pre-judge whether the representations involved in a physical computation must also play the same causal roles (e.g. whether they have propositional structure, are accessible to consciousness, capable of driving our behaviour in certain ways, and so on). These are important questions, but they are posterior to the question of whether physical computation involves representational content of any kind at all. In what follows, I will phrase the debate in terms of representational content. Roughly speaking, a *representation* need support no more than a basic notion of aboutness or reference. A representation should link an entity and a content, such that the entity represents its content. Nothing more is required. In particular, it is not required that any of the conditions above concerning intentional content are satisfied.

Third, the dispute over the received view is sometimes conducted in terms of the nature and individuation of physical computational *states*. As I will argue in Section 2.3, questions about individuation of physical computations should be phrased in terms of *processes*: the basic units of physical computation are computational processes, and individuation of physical computational states is parasitic on the individuation of computational processes. This is more than just a matter of book-keeping. As argued in Section 2.3, excessive focus on computational states has led to the unjustified assumption that physical computations could only essentially involve narrow content.

Finally, in what follows, 'computational process', 'physical computation', and 'computational system' will be used interchangeably. What is meant is the implementation of a computation by a physical system.

---

3. Burge (1986); Egan (1991, 1992, 1994, 1995); Fodor (1980); Segal (1989); Shagrir (2001).

## 2 Arguments against the representation condition

This section presents three arguments against physical computation necessarily involving representation. The first argument has few explicit defenders, but it is worth considering because it still has a significant influence in the philosophical literature. The other two arguments are carefully developed by Egan and warrant more attention.

### 2.1 Distinction between dynamics and individuation

A seductive line of thought about physical computation appears to argue against the received view. Computations are formal, their transitions are governed only by the syntactic character or ‘shape’ of the computational tokens. A computational token’s shape typically covaries with its semantic properties. But it is the shape that does the causal work in the transitions, not the semantic properties. Indeed, computation is a way in which a physical process can appear to be semantically sensitive without spookily depending on what its tokens represent. Therefore, semantic properties do no essential causal work in the transitions of computational processes. The conclusion is that semantic properties ride along with computations, but they are not among their essential properties. Therefore, representational content is not essential to computation.<sup>4</sup>

Three problems should be noted with this argument.

First, the argument mistakes the form of dependence on representational content at issue in the received view. An advocate of R does not claim that the *causal dynamics* of computations depend on representational content. Her claim is that the *individuation* of computations depends on their representational content. The battleground for the received view are the facts about the individuation of computations, not the facts about their dynamics. If one cannot make sense of computational identity, or a computation/non-computation contrast, without reference to representational features, then representation is an essential feature of computation, regardless of its claimed irrelevance to causal dynamics. A non-semantic nature for computational *transitions* is compatible with the claim that computation essentially involves representational content.

A second problem is that the argument relies on a questionable notion of ‘essential causal work’. It is not obvious that if one property (or cluster of properties) ‘does the causal work’ of another, then that latter property is causally irrelevant. As debates over Kim’s exclusion argument illustrate, to assume such a principle is valid commits oneself to a strong form of reductive materialism where the only causally relevant properties lie at the bottom level of physics, if such a level exists. Unless one has no qualms about such a reduction, the argument above that semantic properties of computations are causally irrelevant because their work is done by syntactic properties should be regarded with suspicion. Further justification is required to show that semantic properties are somehow specially disreputable.

---

4. Cf. Searle (1990)’s ‘axiomatic argument’ against Strong AI: since computational operations are purely formal, computations need not have semantic content. And Stich (1983)’s argument that the computational theory of mind supports eliminativism about intentional content because, if mental processes are computations, their representational content is explanatorily irrelevant to their causal dynamics (p. 193).

Finally, *pace* Kim's argument, if one is willing to grant computational states causal powers at all, there are positive reasons for thinking that connections exist between the facts about individuation and the facts about causal dynamics. Facts about causal dynamics include, *inter alia*, facts about which *events* are involved in the causal dynamics. The events that exist, and whether those events include the tokening of computational states, depend on facts about the individuation of computations. If one wishes to allow computational states to have causal powers *qua* computational states, then appeal to their individuating properties is required to account for the causal dynamics of the system. If those individuating properties include their semantic properties, then their semantic properties will essentially figure in the causal dynamics. Therefore, although computational transitions may not require any spooky dependence on semantic properties, that does not mean that semantic properties are causally irrelevant in the overall causal dynamics of the system. If computational states *qua* computational states have causal power, then facts about their individuation cannot be separated from the facts about their causal dynamics.

## 2.2 Egan's argument from interpretation of Marr

The debate over whether computation essentially involves representational content has centred around the interpretation of Marr's theory of vision.

A number of participants defend the view that Marr's theory of vision makes essential use of representational content.<sup>5</sup> Egan (1995) argues that this is wrong. She claims that although a *complete* explanation of visual processes will typically advert to representational content, the *computational core* of such a theory is purely non-representational and mathematical. Just as an explanation of the ability of a sand shark to detect its prey using electric fields adverts to both electromagnetic theory and the fact that in the shark's environment, animals, but not rocks, produce significant electric fields, without the latter environmental fact being part of *electromagnetic theory*, so an explanation of vision will typically appeal to representational features beyond purely formal computational properties, without those representational features being part of the computational theory of vision.

What reason does Egan give for thinking that computational theories of vision are non-representational and mathematical? She starts by drawing attention to Marr's three levels of computational description: (i) the *computational* level, which characterises the function computed by the system; (ii) the *algorithmic* level, which specifies an algorithm for computing that function; and (iii) the *implementation* level, which describes how the process is physically realised. Marr's computational level is sometimes thought to be intentional and occasionally straightforwardly equated with Pylyshyn (1984)'s semantic level, Newell (1982)'s knowledge level, or Dennett (1987)'s intentional level. Egan claims that this is a mistake. To adopt Marr's computational level is not to adopt any kind of intentional or semantic description, but instead the point of view of mathematical computation theory for describing the system. In the context of Marr's theory, the computational level should be understood in a *mathematical function-theoretic* way of describing the system. Computational description is a characterisation of the functions computed by the various parts

---

5. The primary concern is to decide whether the representational content is wide or narrow. Burge (1986); Davies (1991); Kitcher (1988); Shapiro (1997) defend broad content, and Cummins (1989); Morton (1993); Segal (1989, 1991) defend narrow content.



of the cognitive system in mathematical terms, not in terms of what those parts represent:

I have argued that from a computational point of view [the retina] signals  $\nabla^2 G * I$  (the X channels) and its time derivative  $\partial/\partial t(\nabla^2 G * I)$  (the y channels). From a computational point of view, this is a precise characterization of what the retina does. Of course, it does a lot more—it transduces the light, allows for a huge dynamic range, has a fovea with interesting characteristics, can be moved around, and so forth. What you accept as a reasonable description of what the retina does depends on your point of view. I personally accept  $\nabla^2 G$  as an adequate description, although I take an unashamedly information-processing point of view. (Marr 1982, p. 337)

At the computational level, the retina computes  $\nabla^2 G$ .  $\nabla^2 G$  is a mathematical function that maps two-dimensional arrays  $I(x, y)$  to isotropic rates of rates of change of  $I(x, y)$  at points  $(x, y)$  via convolution. A computational description of the visual system is a purely formal mathematical description of the function computed by the system. It is not a description of the visual system in terms of, say, representations of *light intensity values* or *shape*. It is neutral about whether the inputs and outputs to the system have any representational content at all.

Egan argues that if Marr's computational level need not involve representational content, then his algorithmic level need not involve representational content either. Therefore, the computational core of Marr's theory makes no use of representational content: it is a purely mathematical construction. Egan acknowledges that Marr thinks that the visual system has other properties—and to the list above she would presumably add representational properties—but those properties are not essential to the computation that the system performs. Hence, a paradigmatic case of a computation performed by a physical system—the computation Marr attributes to the human retina—does not essentially involve representational content. Therefore, R is false.

What can be said in response?

First, a distinction should be drawn between mathematical computation theory and Marr's computational level. Mathematical computation theory is a branch of pure mathematics and concerns relations between mathematical structures and objects. The 'computers' it studies are mathematical entities, not physical systems. According to mathematical computation theory, a Turing machine is not a physical system; it does not 'perform' a computation in the sense that a physical system does. A Turing machine is typically identified with a set of set of mathematical symbols, e.g. with the quintuple  $M = (Q, \Sigma, \Gamma, \delta, q_0)$ , where  $Q$  is a set of state symbols,  $\Gamma$  is a set of numerals that can be used on the tape,  $B$  a special symbol that represents a blank,  $\Sigma$  a subset of  $\Gamma - \{B\}$  called the input alphabet,  $\delta$  is a partial function from  $Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$  labelled the transition table,  $q_0 \in Q$  a special state called the start state, and the symbols  $L, R$  the direction of movement along the tape. The Turing machine's tape is another mathematical entity in which symbols are kept in a linear order.<sup>6</sup> One can make sense of a Turing machine having 'inputs': the initial symbols on the tape. One can make sense of a Turing machine having 'outputs': the final

6. See Sudkamp (1998), pp. 259–260 for more on the definition of a Turing machine.

symbols on the tape. But in each case the inputs and outputs are mathematical objects, typically, ordered sequences of numerals. It should be emphasised that these inputs and outputs are not empirical *ink-marks*. They are mathematical entities, numerals understood as abstract objects. Typically, one does not care what these abstract entities are, so long as one can make sense of them being numerically distinct. The computations studied in mathematical computation theory are independent of how things are in the physical world. They are independent of empirical ink-marks, they do not ‘take place’ in time, or depend on the physical possibility of infinitely long tapes. Computers in mathematical computation theory are mathematical entities that bear certain relations, studied by that theory, to other mathematical entities, the functions they compute.

Likewise, Euclidean geometry, by itself, says nothing about the structure of the physical world. Of course, thinking about physical triangles, lines, and planes often has a heuristic and propaedeutic value when thinking about Euclidean geometry. Similarly, thinking about physical paper-tape machines often has a heuristic and propaedeutic value when studying mathematical computation theory. Secondly, a significant motivation for studying these areas of pure mathematics, and the reason why they are classified as distinctively ‘computational’ or ‘geometrical’ is their potential application to physical systems. However, it should be clear that physical entities are *not* the subject matter of the relevant mathematical claims. Mathematical computation theory does not say anything about physical systems.

Consequently, mathematical computation theory does not, by itself, have the resources to explain how the visual system works. Therefore, Marr’s computational level cannot straightforwardly be identified with the function-theoretic descriptions given in mathematical computation theory. Marr needs some way of connecting the abstract mathematical descriptions to the nuts and bolts of physical reality.<sup>7</sup> How do mathematical entities, like  $I(x, y)$  and  $(x, y)$ , get connected to the human visual system? One possible answer is the *realization function*:

... a realization function  $f_R$  ... maps equivalence classes of physical features of a system to what we might call “symbolic” features. Formal [computational] operations are just those physical operations that are differentially sensitive to the aspects of symbolic expressions that under the realization function  $f_R$  are specified as formal features. The mapping  $f_R$  allows a causal sequence of physical state transitions to be interpreted as a *computation* ... Given this method of individuating computational states, two systems performing the same operations over the same symbol structures are computationally indistinguishable. (Egan 1992, p. 446)

A realization function maps physical nuts and bolts to the formal symbols employed by computation theory. It is the link between the abstract world of mathematical computation and the empirical world of the human visual system. At least according to Egan’s earlier

---

7. Note that this is different from the question that Egan (1995) considers on pp. 189–194: how a formal computational account of the visual system can connect to explaining the *intentional* capacities of the subject. The question here is how a formal computational description can even *be about* the human visual system.



work, a realization function does the bulk of the work of computational individuation; it determines whether a physical system performs a computation, and if so, which computation it performs.<sup>8</sup> If a realization function obtains between a physical system and the abstract entities involved in a mathematical computation, such that transitions in the physical system mirror those of the symbolic entities in the mathematical computation, then *that physical system implements that computation*. A realization function is to be understood non-semantically: it associates physical states with mathematical entities independently of any representational content that those physical states might have. A realization function is no more than a mapping (a pairing) between classes of physical features and abstract entities.

Egan (this volume) clarifies the role of the realization function: physical computational identity depends *both* on a realization function and a special kind of semantic interpretation that assigns exclusively mathematical content to physical states. Appeal to *distal environmental* content is banned in establishing computational identity. To the extent that Egan asserts that the relationship between physical states and mathematical entities has to be one of representation, I agree. I argue against her ban on distal representational content in Section 2.3. However, it is worth considering, as initially appeared to be in the offing, whether one can have computational implementation only with a realization function, i.e. without a commitment to representational content of any kind.

One problem that a non-semantic view faces is the threat of universal realization. A realization function is a mapping between a mathematical computation and a physical system, and such mappings are cheap. There are billions of particles in a brick wall undergoing complex patterns of activity (gravitational, thermal, electromagnetic, etc.). Searle (1992) claims that there are so many physical patterns inside a brick wall that there is almost certain to be at least one pattern that maps onto the structure of any finite mathematical computation one likes. Putnam (1988) shows that one does not need a complex system like a brick wall to have universal realization on this view of computational implementation. By just considering the state of a single particle as it evolves over time one can construct a realization function that makes that particle perform any finite computation one likes. Chalmers (1996) argues that, with some qualifications, the transitions involved can also support counterfactuals.<sup>9</sup> If the mere *existence* of a realization function is sufficient for a physical system to perform a computation, then almost every physical system performs every computation.

If Egan's realization functions determine computational identity, then the computational identity of physical systems would be trivial. In order to avoid this, some extra constraints are required. Not every realization function should establish a computational identity.

8. '... the fact remains that the realization function  $f_R$  determines how the computational states are individuated. This function individuates computational states *non-semantically*, that is, independently of any particular semantic interpretation such states may have ... while the semantic interpretation does provide a useful description of what the system does, it does not serve to *individuate* the underlying computational states.' (Egan 1992, p. 448); 'While the interpretation function  $F_I$  plays an important explanatory role in a computational theory, it does not, however, play an individuating role. That is the job of  $F_R$  [the realization function].' (Egan 1994, p. 261).

9. Putnam (1988)'s argument is made in terms of finite state automata but can be applied to other computational formalisms. Chalmers (1996)'s argument requires that a physical system possess a 'clock' and a 'dial', but these conditions are easily satisfied.

But what makes certain realization functions special? Why does the existence of some, but not other, realization functions suffice to establish a physical computation?

Two tempting answers should be avoided.<sup>10</sup> First, it cannot be that some realization functions do not establish a physical computation because they are somehow insufficiently ‘direct’ or ‘too disjunctive’. There are plenty of disjunctive realization functions that do establish physical computations. Think of the kinds of realization functions involved in electronic PCs. There is nothing direct or natural about mapping the voltage levels  $5 \pm 0.4$  V in a 01100001 pattern in certain capacitors scattered throughout a machine to the symbol ‘a’. Such a mapping is not perspicuous independent of familiarity with its practice, nor does it cut the world at natural joints. Yet it is the realization function employed every day in electronic computers. Unless one denies that those systems perform computations, one cannot rule out a realization function because it is non-natural or too disjunctive.

Second, the reason why some realization functions are objectionable cannot be that it requires excessive epistemic work on the part of the agent to construct those realization functions.<sup>11</sup> First, as noted above, the realization functions of electronic PCs often require considerable epistemic work on the part of a cognitive agent to construct. Sometimes the cognitive agent (the hardware designer) has to perform the computation herself in advance in order to construct the relevant realization function. Second, this response makes the computation a physical system performs entirely a function of the epistemic powers and interests of cognitive agents. It entails a form of anti-realism about computation: the facts about computation are not constrained by the system itself (which trivially perform every computation), but only by the facts about epistemic agents investigating that system. Unless one wishes to endorse a strong form of anti-realism about computation, questions about how a realization function is found or constructed should not matter when establishing computational identity. What matters is why the *existence* of certain realization functions, independent of how we discover them, is sufficient to establish computational identity.

One might attempt to avoid the threat of universal realization by claiming that Marr did not say that the visual system performs a particular computation *simpliciter*, but only that it performs a particular computation *under a particular realization function*,  $F_R$ . This would block the worry that Marr’s claims would be trivially true. However, it would go beyond appeal to the mere existence of a realization function to explain how Marr’s theory relates to the visual system. It would be to say that not only does that realization function exist, but that it is also somehow special, or particularly relevant, to explaining the human visual system. But in what does the relevance of that particular realization function lie? Why is the computation that the retina performs  $\nabla^2 G$ , and not something else (as it would be under any of the other realization functions that are satisfied by the human retina)? This is not a question that a realization-based account of computational identity has the resources to answer.

However, a natural answer to this question lies at hand. What makes a particular pairing

---

10. Variations on these answers are explored and endorsed in Cummins (1989), Ch. 8 in the context of ruling out unintended interpretation functions. See also the discussion in Egan (1992), pp. 450–451; Egan (1995), pp. 192–193.

11. Cummins (1989) appears to endorse this as part of his condition on an interpretation function (pp. 102–105).

between the nuts and bolts of an empirical system and mathematical entities (e.g. numerals or numbers) special is that those nuts and bolts *represent* those entities. Certain realization functions are special because they truly describe representation relations in the world. The retina performs the computation that Marr suggests because its inputs and outputs *represent* the relevant mathematical structures. Representation is the crucial link between the mathematical and empirical world. It draws the line between relevant and irrelevant realization functions and winnows down the infinity of pairings in the only way that respects the flexibility of implementation of computation.

As noted above, Egan (this volume) now emphasises that certain mappings between physical structures and mathematical structures are special because the relevant physical states represent mathematical entities. Egan claims that this representation relation, called the *canonical description* or *interpretation function*,  $f_I$ , together with the non-semantic realization function, jointly determine computational identity. A puzzle remains on this view. Once the representational  $f_I$  is admitted, it is unclear what role is left for the non-semantic realization function  $f_R$  to play. If two physical systems map, via their interpretation functions, onto the same abstract mathematical computation, then that seems both necessary and sufficient for those two systems to perform that computation. Similarly, if their interpretation functions map two physical systems onto different mathematical computations that appears both necessary and sufficient for them to perform different computations. All the work in establishing computational identity is done by the representational mapping,  $f_I$ , with nothing left over for  $f_R$  to do. Effectively, the relationship between the nuts and bolts and mathematical entities that  $f_R$  purports to describe as special, if sense can be made of it that is non-trivial, appears to be already entirely fixed by the representation relations described by  $f_I$ .

### 2.3 Egan's dilemma concerning narrow content

Egan's second argument against R involves an ingenious dilemma concerning narrow content.<sup>12</sup> Egan argues that the nature of physical computation forces advocates of computational psychology into a dilemma. Either they can hold onto the received view but be committed to all cognitive computations involving narrow content (as Cummins (1989); Fodor (1980); Segal (1989) do), or they could give up the received view entirely. Egan argues that we should choose the latter option, for two reasons. First, it is notoriously difficult to make sense of a notion of narrow content. Second, it is hard to see how, even if a notion of narrow content could be constructed, it would live up to its billing as a form of representational content that is relevant to psychology, where content is typically specified in broad, environment-specific, terms. Computational theories like Marr's often ascribe broad content to their processes (*surface orientation*, *depth*, etc.), not narrow content.<sup>13</sup>

Egan's argument is based on the assumption that the only kind of representational content that computations could essentially involve is narrow content. Since narrow content is

---

12. Narrow content supervenes on the intrinsic physical state of the agent, i.e. content that any intrinsic physical duplicate would share irrespective of its external environment. Broad content fails to satisfy this condition.

13. Egan (1995), pp. 194–195.

either unavailable, or a bad fit with the practice of computational psychology, we should reject R. In short, the cost of R is an unacceptable commitment to narrow content.

Egan's argument works only if:

- N1 The only kind of representational content that computations could essentially involve is narrow content
- N2 A restriction to ascription of narrow content is intolerable in computational psychology

I will argue against N1: there is no reason why physical computations in psychology, or elsewhere, cannot essentially involve broad content. Cummins (1989); Fodor (1980, 1987); Segal (1989, 1991) have argued against N2 and for the workability of narrow content in computational psychology. I believe that we do not face such a dilemma. There is no reason why representational content in computational psychology cannot be either broad or narrow.<sup>14</sup>

What is Egan's argument for N1?

Egan claims that unless computational individuation is narrow, some important scientific generalisations are lost. According to Egan, a computational description is a description of a mechanism without reference to its environment. The environment-independence of this description allows us to make sense of a mechanism being well or ill adapted to various environments. On the other hand, if computational descriptions were environment-dependent, then the same computational mechanism could not be freely considered in different environments. One could not make sense of apparently legitimate scientific questions about *the same mechanism* being well or ill adapted in other environments.

It is precisely because a computational theory provides an environment-independent characterization of a device that we can see why *this* mechanism would not have been adaptive if the environment had been different, and why it might cease to be adaptive if the environment changes.

(Egan 1994, p. 264)<sup>15</sup>

For example, consider a subject for whom Marr's theory provides a correct account of her visual processes. Suppose that this individual were transported to an environment where her perceptual states have a different content. If computational identity essentially depends on broad content, then the computational identity of her visual processes would differ. Suppose in our environment, the computation that the subject performs is successful at

---

14. Wilson (1994) argues that the vehicles of computations in psychology can extend outside an individual's head and include objects in his or her environment (cf. Clark and Chalmers (1998)). This claim should be distinguished from N1, which concerns the *content* of the representations involved in computations. As noted by Segal (1997), Wilson's anti-individualism about vehicles is compatible with N1 about content. Wilson's view only introduces dependence on objects in the subject's nearby environment. Broad content can introduce dependence on environmental objects that are spatially and temporally distant (e.g. originally-dubbed water samples).

15. Also Egan (1991), pp. 199–202; Egan (1992), p. 447.

recovering visual information from the environment. It seems coherent to ask whether, in the other environment, *the same computation* would be successful at recovering visual information. But if broad content determines computational identity, then this latter question cannot be asked. In the other environment, the computation would simply not occur; the computational identity of the visual system would be different.<sup>16</sup> This appears to render incoherent apparently legitimate questions about whether *the same computational system* is well or ill adapted in different environments.

What can be said in response?

Talk of adaptation can be recovered using broad content with a combination of two strategies.

First, even if the computational nature of a device changes across different environments, one can still make sense of the same *physical device* being well or ill adapted to an environment. One can interpret talk of whether the human visual system is well or ill adapted to an environment as talk of whether the physical system associated with the human visual system is well or ill adapted to that environment. If the physical system associated with the human visual system in the actual world performs poorly at recovering information in a different environment, then the computation performed by the human visual system is poorly adapted to that different environment. If the physical system performs well at recovering information from the different environment, then the computation performed by the human visual system is well adapted to that environment. Either claim can be true, and explanatory, even if the computation performed in the new environment is different from that performed in the actual world. On this strategy, claims about adaptation of computational mechanisms are understood as claims about adaptation of the physical system implementing the computation in the actual world.

A second strategy is that, when considering a counterfactual scenario, one may *stipulate* representational content as part of the supposition. A physical duplicate might, in virtue of being in another environment, have a different representational content and so a different computational identity. However, that does not stop us from using that physical duplicate to perform the same computation as in the actual world by setting up appropriate representational conventions. Representational conventions are easy to set up: they can be created by stipulation. There is no reason why a physical system cannot have more than one kind of representational content associated with it. It may have broad content, acquired in virtue of its relation to its environment, and stipulated content, acquired in virtue of the suppositions under which we consider the scenario. Claims about how a computation in the actual world performs in a different environment can be understood as claims about the performance of a physical duplicate in that different environment *where the system is interpreted so as to perform that same computation* (i.e. where we adopt appropriate representational conventions as part of our counterfactual imagining). In other words, the question we entertain when considering adaptation under this strategy is: supposing that a physical duplicate were to perform *that same computation* in the counterfactual environment, how successful would it be in that environment?

---

16. This is not because the visual system could receive different *input* in the other environment—a difference in input does not amount to a difference in a system's computational identity. The worry is that the *computational mechanism* that operates on inputs would differ in the different environment.



When thinking about the degree of adaptation of a computational mechanism in different environments, we may flip between either strategy. Sometimes we acknowledge that in a different environment the physical system performs a different computation in virtue of having different representational content (strategy 1). Sometimes we consider the physical system in such a way as to force it to perform the same computation regardless of its environment (strategy 2).

The cases discussed by Burge (1986) highlight the divergent nature of our judgements about computational identity across different environments. Burge describes the human visual system as performing an adaptive *crack*-processing computation in a crack-based environment, and a physical duplicate of the organism as performing an adaptive *shadow*-processing computation in another shadow-based environment.<sup>17</sup> The physical duplicates have different broad contents, and therefore on Burge's view, perform different computations. The former embodies assumptions about *cracks*. The latter embodies assumptions about *shadows*. However, we can make sense of two, apparently competing, judgements about adaptation of the human visual system in the two environments. On the one hand, we can make sense of the judgement that *the computation performed by the human visual system is adaptive* in both environments. Under strategy 1, one can see that the same physical system is successful in both environments. The computation performed by the physical system is adaptive in each case, even if the computations performed in each case are different. On the other hand, one can make sense of the judgement that *each computation is adaptive only in its own environment*. Under strategy 2, one can see each system as performing the same computation across both environments by representational stipulation. By stipulating that the states in the second system represent *cracks*, we can see that the first, *crack*-based, computation is poor at detecting cracks in the second environment: it would embody false assumptions about cracks. And one can see that the second, *shadow*-based, computation is poor at detecting shadows in the first environment: it would embody false assumptions about shadows.

Switching our attention between the computation determined by broad content and the computation determined by representational stipulation is part and parcel of our thinking about computation across different environments. The combination of the two strategies allows one to accommodate the competing judgements that in one sense (made clear by strategy 1), the computation performed by the visual system is adaptive in both environments, and in another sense (made clear by strategy 2), the computation performed in each environment is adaptive only in its own environment. Both claims are compelling, and understood right, are correct. The two-pronged described strategy above is able to make sense of our judgements about adaptation better than a simple restriction to narrow content. Its availability also shows that one is not forced to adopt N1 in order to make sense of claims about adaptation.

A second influential argument for N1 derives from Fodor (1980). It involves what Fodor calls the *formality condition*. According to Fodor there is something about the nature of physical computation that restricts computations to narrow content. Fodor's exact argument is hard to pin down. I will try to develop it below.

The motivation for the formality condition is to make sense of the ability of physical

---

17. Burge (1986), pp. 39–43.



computations to appear to be semantically sensitive without spookily having access to what their states refer to. Computations are formal: their transitions are governed by the syntactic character or ‘shape’ of their states. A computation can give the appearance of being semantically sensitive if the formal character of its states covaries with its representational content. By tracking formal properties, a computation can appear to track semantic properties. This use of syntactic properties as proxies for semantic properties motivates Fodor’s ‘formality condition’ on computation. The formality condition is that any physical computation that appears to be semantically sensitive should have all relevant semantic distinctions mirrored in formal differences among its states:

the computational theory of mind requires that two thoughts can be distinct in content only if they can be identified with relations to formally distinct representations. (Fodor 1980, p. 486)

Broad content dramatically fails to satisfy this condition. States in two physically identical individuals can be formally type-identical—they can be intrinsic duplicates—and yet have different broad contents. Narrow content appears to be the only way of satisfying the formality condition:

Narrow psychological states are those individuated in light of the formality condition; viz., without reference to such semantic properties as truth and reference. And honoring the formality condition is part and parcel of [computational psychology]. (p. 495)

Broad content appears to reintroduce the spooky kind of dependence on content that the notion of computation was intended to eliminate. Suppose one chooses to individuate computational states in terms of their broad content. At the very least, a physical computation should be consistent in how it handles its representational content: it should consistently map the same representational content to the same representational content. (If a physical computation is not consistent in this, then it makes no sense to individuate physical computations in terms of how they handle their representational content). But how can a computation consistently process broad content without being spookily sensitive to its referent? How can it know that some tokens refer to *water* and others refer to *twater* when those tokens are physically identical? The only type of content that physical computations can consistently process—and hence the only kind of content relevant to the individuation of physical computations—appears to be narrow content.

What can be said in response?

First, it is worth observing that the basic units of physical computation are processes, not states. A physical state counts as computational only to the extent that it participates in a physical computational process. It makes no sense to posit a computational state in isolation from any computational process. Physical computational states must have ‘owners’, computational processes of which they are a part (just as digestion states must have owners, processes of which they are a part). Furthermore, physical computational states are not individuated in isolation. They are individuated by reference to the physical computational process in which they occur. A 5 V signal in one process may play the role

of a halting state, and 0 V the role of a starting state, while a physically identical 5 V signal in another process may play the role of a starting state, and 0 V the role of a halting state. One cannot identify a state as *starting* or *halting* without reference to the process in which it occurs.

The motivation for the formality condition is to make sense of the apparent semantic sensitivity of computations. However, what is required to account for this is that the relevant semantic distinctions between *a process's* states should be mirrored in formal differences among those states. It is not also required that all semantic properties, or semantic differences between the states of that computational process and those of any other computational processes, should be so marked. Only the semantic differences to which a computational process appears to be sensitive need be formally encoded by its states.

For example, suppose a computation appears to distinguish input tokens in English that represent *fire* from those that represent *water*. If presented with an input token that represents *fire*, the computation outputs the string of characters 'fire', and if presented with an input token that represents *water*, it outputs the string 'water'. Such a computation is a simple example of apparent semantic sensitivity. How could such a computation work? Unless it is spookily sensitive to its referents, it works by the relevant semantic differences between its tokens being mirrored in formal differences. The input tokens must have some formal difference that distinguishes tokens that represent *fire* from those that represent *water*. The tokens need not encode any other differences than those required to explain the behaviour above. For example, it is not necessary that a formal difference between tokens that represent *fire* and *chalk* be marked to explain the semantic sensitivity above. The formal structure of the tokens need only encode the information that *fire* and *water* tokens are relevantly different tokens. It need not encode all their semantic properties. In particular, it need not be sufficient to fix their referents: it need not be sufficient to determine that *fire* tokens refer to *fire*, and that *water* tokens refer to *water*, and rule out that the respective tokens refer to, say, 0 and 1, or *chalk* and *cheese*, or any other pair of distinct contents.

There is no reason, at least none stemming from explanation of apparent semantic sensitivity, to think that the formal structure of a computational state should determine its representational content. Therefore, there is no reason why its representational content should supervene on its formal properties. The formal structure is only needed for the computation to keep track of the semantic differences relevant to the process, not to encode all semantic properties. Consequently, there is no reason why the contents of computational tokens cannot be broad. So long as a difference between the *fire* and *water* tokens is marked, it does not matter whether their respective content is broad or narrow.

One might object that broad content is still difficult to process. If computations are individuated in terms of their representational content, then a computation should be consistent in how it processes its representational content: it should consistently map the same representational content to the same representational content. Broad content appears to interfere with this condition. Two computational tokens may have different broad content and yet be physically identical. Such tokens would appear apt to disrupt the consistent handling of representational content. What is to stop a computation that ostensibly produces a consistent *water-water* mapping from, on any given occasion, pro-

ducing a spurious *water*–*twater* mapping, or any other *water*–*X* mapping, where *X* is physically identical to a *water* token but with different broad content? The computation cannot distinguish between *water*, *twater*, and *X* tokens, so there seems no way in which it can consistently respond to, and yield, all and only *water* tokens. Any *water* processing computation appears vulnerable to being hijacked by doppelgangers with different broad content. Therefore, computations over broad content cannot establish consistent processing relationships between representations, and hence cannot establish consistent computational identity.

However, the worry is unfounded. Cases of broad content involve some form of environment dependency. Two physically identical tokens can have different content (*water* and *twater*) because they exist in different environments. Within a single environment, broad content is the same (e.g. every token ‘water’ refers to *water*). The worry raised above concerns whether a computation can consistently process broad content. But a given computational process occurs in a particular environment, and throughout that process, the environment fixes unambiguous broad content for the tokens.<sup>18</sup> A problematic computational process described above—containing a mixture of physically identical *water* and *twater* tokens—is not a possibility supported by the arguments for broad content. As far as broad content is concerned, tokens within a process are either all *water* tokens or all *twater* tokens depending on the environment in which the process is located. Therefore, the comparison relevant for whether broad content can be consistently processed is between tokens that could take place *in the same environment*, not tokens that could take place anywhere in any environment. Within an environment—and a physical computational process always occurs in some environment—there is no reason why computations cannot consistently process broad content. Broad content does not threaten the consistent handling of representations by computational processes. There is nothing wrong with saying that my Twin Earth counterpart possesses an information-processing subsystem that reliably processes *twater* tokens, while I possess a reliable *water* processor. There appears no reason why a computation within an environment cannot process broad content just as consistently as it could narrow content.

Fodor’s formality condition on computation is:

- F1 Any difference in representational content between computational states should be mirrored in a difference in their formal structure.

But a weaker condition suffices to make sense of semantic sensitivity:

---

18. Cases where the computational process is transported between environments during the computation, or where the environment changes during the computation, can also be handled. They should be treated in the same way as cases of individuals transported from Earth to Twin Earth. The thought is that the individual would continue having Earthly content for some time after the transportation (certainly enough time to complete any thoughts), and only gradually acquire Twin Earth content as he or she becomes embedded in Twin Earth representational relations and conventions (Block 1990). Hence, one should not expect inconsistent information processing. Similarly, cases in which a process is so large as to span two environments pose no problem, since the process can be consistent in how it handles broad content within each part of the process, and this is sufficient for consistent individuation.

- F2** For a computational process,  $P$ , all differences in representational content to which  $P$  appears to be sensitive should be mirrored in a formal difference between  $P$ 's states.

F2 is compatible with the representational content being broad or narrow.

Why does Fodor endorse F1 rather than F2? The comparison class one has in mind when explaining semantic sensitivity is crucial. If the comparison class includes all possible physical computational states in any environment, then every aspect of representational content should supervene on formal structure, and F1 follows. However, as should be clear, the comparison class is smaller, namely, the class of tokens that could occur in the computational state's 'owner' process. If the link between computational states and their owners is ignored, then an incorrect comparison class is introduced. With that connection in mind however, F1 can be replaced with F2 with no loss to explanations of semantic sensitivity.

A consequence of F2 is that there are no interesting constraints, stemming from the nature of computation, on whether computational psychology is individualistic. *Contra* Fodor and Egan, computational psychology need not be individualistic. This raises a question: given that there are no principled constraints stemming from the nature of computation, are there any constraints from the practice of computational psychology that determine whether computational psychology is individualistic? Egan (1991) argues that when this question is asked about psychology as a whole, there is no single correct answer. In some cases, psychological explanations attribute narrow content, such as Marr's ascription of representations of proximal features of subject's visual field, e.g. *blobs*, *virtual lines*, and *zero-crossings*. In other cases, psychological explanations attribute broad content, such as Marr's ascription of representations of distal features, for example, *surfaces*, *shadows*, and *cracks*. On the version of the received view described above, one should expect a similarly mixed answer to the question of whether computational psychology is individualistic. There is no reason why computational psychology should be restricted to narrow content.

### 3 Arguments for the representation condition

Let us turn to the arguments for R. A computation maps certain inputs to certain outputs. A physical computation is a mapping between real-world *stuff*: it takes stuff (ink-marks, electrical signals, etc.) as input and yields other stuff, or other arrangements of stuff, as output. The claim defended below is that computations are not just mappings between any kind of stuff, but mappings between *stuff that represents*.

Here are three arguments for why computation has to involve representational content.

#### 3.1 Paradigmatic cases of computation involve representation

Many paradigmatic cases of computation involve representation. For example, Turing's mindless clerk, who performs computations by hand, performs a mapping between

representations.<sup>19</sup> The clerk maps representations (ink-marks on the page) to other representations (other ink-marks on the page). The clerk's ink-marks can be interpreted as representing either numerals or numbers. This ambiguity is not unusual. The ink-marks, 1, can represent either the numeral '1' or the number 1, depending on context. The context may be specified by adding quotation marks, but this convention is not always decisive.<sup>20</sup> The same physical stuff may also have multiple representational contents associated with it, and consequently the same physical process may have multiple computational identities.

Another paradigmatic case, electronic computation, also involves representation. An electronic computer takes electrical signals as input and yields electrical signals as output. The input and output electrical signals of a computer represent. Typically, the electrical signals of a computer represent 0's and 1's. Again, there is room for multiple representation. A given signal may represent both a long sequence of 0's and 1's and the text of a new e-mail message. Or, a given signal may represent both a sequence of 0's and 1's and a picture to display on the screen. Algorithms manipulate signals that represent. A sorting algorithm manipulates electrical signals that represent *letters* or *sentences*; a compiler manipulates electrical signals that represent formal states of another machine; a syntax checker manipulates electrical signals that represent well or ill formed strings of letters.

The fact that paradigmatic cases of computation involve representation is far from conclusive as an argument for R, but it is not negligible either. When questions arise as to whether a case counts as a computation or not, it is salutary to keep the paradigmatic cases in mind. It is also worth observing the marked lack of paradigmatic cases of computation without representational content.

### 3.2 Representation is involved in the notion of I/O equivalence

Our notion of computation involves a notion of input–output (I/O) equivalence. We sometimes claim that two physical systems are 'computationally equivalent' or 'compute the same function'. What we mean is that the two systems perform the same computational task, even if they use different computational methods to achieve the same ends. I/O equivalence is a necessary, but not a sufficient, condition for computational identity.<sup>21</sup> I wish to argue one cannot make sense of I/O equivalence without requiring that computation involves representational content.

Imagine two I/O equivalent systems that are made out of different physical materials. One system is made out of silicon and takes electrical signals as inputs and outputs, the other system is made out of tin cans and string and takes marbles as inputs and outputs. Suppose that the two systems are computationally I/O equivalent. What could their I/O equivalence consist in? The respective inputs and outputs of the two systems are different, and may be so different as to not have any physical or functional properties in common.

---

19. For a description of why human computers are paradigmatic cases of computation, see Gandy (1988); Sieg (1994, 2001).

20. That there is an ambiguity about *content* seems required to make sense of the possibility of a use/mention confusion. What is at stake in a disagreement about use/mention is whether one is talking about numerals or numbers when employing the ink-marks.

21. See Sprevak (2007).

The only answer seems to be that their respective inputs and outputs *represent the same thing*.

Consider two computational systems that perform the same numerical calculation. Suppose that one system takes ink-marks shaped like Roman numerals (I, II, III, IV, ...) as input and yields ink-marks shaped like Roman numerals as output. Suppose that the other system takes ink-marks shaped like Arabic numerals (1, 2, 3, 4, ...) as input and yields ink-marks shaped like Arabic numerals as output. Suppose that the two systems compute the same function, say, the addition function. What could their I/O computational equivalence consist in? Again, there may be no physical or functional identity between their respective inputs and outputs. The only way in which their inputs and outputs are relevantly similar seems to be that their inputs and outputs represent the same thing.

Egan claims that the inputs and outputs of computations can be characterised purely functionally in a way that does not appeal to representational content:

To describe something as a symbol is to imply that it is semantically interpretable, but (and this is the important point) its type identity as a symbol is independent of any particular semantic interpretation it might have. Symbols are just functionally characterized objects whose individuation conditions are specified by a realization function  $f_R$ . (Egan 1992, p. 446)

Similarly, Piccinini claims that one can make sense of facts about the I/O equivalence of computations using functionally characterised symbols and without appeal to their representational content.<sup>22</sup> The problem is that there do not seem to be sufficiently wide-ranging non-semantic functional characterisations to capture all the relevant facts about computational equivalence. Given the huge diversity of physical and functional properties of systems that are computationally equivalent, restricting attention to non-semantic resources cannot account for the facts. The only thing that two physically diverse inputs and outputs have in common is that they represent the same thing.

Appeal to representational content also recovers intuitions about I/O equivalence that those hostile to R claim as uniquely their own. Consider the following case. Two physical processes that are intrinsic physical duplicates may have different representational contents associated with them, and hence different computational identities. One physical process may calculate chess moves, while a physical duplicate of that process calculates stock market predictions. We seem inclined to say that, in a sense, the two processes compute different functions, yet in another sense they are I/O equivalent. Appeal to representational content can accommodate both judgements.

The 'not I/O equivalent' judgement is straightforwardly entailed by the representational account above: the two processes are not I/O equivalent because they map different representational content to different representational content (chess moves in one case, stock prices in another). It is less obvious how the account can recover intuitions about the two physical duplicates being, in a sense, I/O equivalent. This can be done as follows. In such cases, we are inclined to judge the two processes as I/O equivalent because we *easily interpret* the two processes so that they compute the same function. When faced with two

---

22. Piccinini (2008), pp. 223–224.



physically (or functionally) identical processes, it is obvious, and highly tempting, to think of them as possessing a common representational content, such that they both map the same content to the same content. Such shared representational content can be assigned via stipulation if need be. The most common way to do this is to interpret the two processes as involving representations of *numbers*, which are independent of the environment in which the process is embedded. It is our disposition to find shared a representational convention when faced with physically identical process that is responsible for our judgement that the processes are computationally equivalent. Notably, what prompts the equivalence judgement is not, as Piccinini and Egan have it, their physical identity, but that their physical identity prompts us to assign a common representational content in addition to whatever other content they may possess.

### 3.3 Representation is needed to make some basic distinctions

Any plausible notion of computation needs to make certain basic distinctions. Failure to make these distinctions marks a failure to give an adequate account of computation. One such basic distinction is between AND gates and OR gates, the building blocks of many computers. An account of computation that fails to capture this distinction cannot be adequate or complete as an account of computation.

AND and OR gates have the following characteristics. The output of an AND gate is *1* just in case both inputs are *1* otherwise it is *0*. The output of an OR gate is *0* just in case both inputs are *0* otherwise it is *1*.<sup>23</sup>

<i>a</i>	<i>b</i>	<i>a</i> AND <i>b</i>	<i>a</i>	<i>b</i>	<i>a</i> OR <i>b</i>
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

Table 1: AND and OR gates

Consider an electrical system with following characteristics. The system gives an output of 5 V if both its inputs are 5 V, otherwise it gives an output of 0 V. Does this system implement an AND gate or an OR gate? At first glance, the system appears to implement AND: it gives an output of 5 V just in case both its first *and* its second inputs are 5 V. But why should 5 V be associated with *1*, and 0 V with *0*, rather than the other way around? If 5 V is associated with *0*, and 0 V with *1*, then the system implements an OR gate. So which gate does the system implement? As the system has been described so far, there is nothing that decides between the two options. No physical, structural, or functional property decides whether 5 V should be paired with *1*, and 0 V with *0*, or 5 V with *0*, and 0 V with *1*. The situation is symmetrical with respect to both assignments.

23. Nothing hangs on the assumption that AND and OR gates involve mapping *truth values* or *numbers*. One might take AND and OR as mapping digit pairs (uninterpreted symbols), such as '0', '1', or '\$', '#'. AND and OR gates will have a different pattern of mapping these formal symbols to formal symbols as per their definitions. The same question will arise: what pattern of formal symbol manipulation does the machine described in Table 2 instantiate?

$in_1$	$in_2$	$out$
0 V	0 V	0 V
0 V	5 V	0 V
5 V	0 V	0 V
5 V	5 V	5 V

Table 2: An implementation of an AND gate or an OR gate?

Appeal to representation allows us to decide between these two options. We can say that *if* an electrical signal of 5 V represents 1, and *if* an electrical signal of 0 V represents 0, *then* the system implements an AND gate. Alternatively, *if* an electrical signal of 0 V represents 1, and *if* an electrical signal of 5 V represents 0, *then* the system implements an OR gate. The difference between an implementation of an AND gate and an OR gate is a difference in representational content.

Notably, appeal to the larger system in which the unit is embedded does not help to determine whether a unit is AND or OR. Irrespective of how large or complex the embedding system, or how many of units obeying Table 2 one strings together, it is still open whether those units instantiate AND or OR. One could adopt a convention where all such units realise AND gates, or all realise OR gates. One could even consistently adopt a mixed convention in which some units, say, those on the left part of the machine implement AND gates, and those on the right part of the machine implement OR. No physical or structural property of the system decides between the options. It is only our representational conventions that settle whether a unit following Table 2 is AND or OR. The temptation to treat a unit satisfying Table 2 as an AND gate derives only from our habit of favouring representational conventions that assign higher numbers to higher voltages, and which are invariant under changes in spatial location.

The representational nature of physical computation is sometimes obscured by the widely accepted claim that computation is *syntactic*. But computation is syntactic in at least two senses. First, as we saw in Section 2.3, computation is sensitive to the formal, syntactic, structure of its input. This requirement is, of course, compatible with the claim that such input has representational content. The second sense in which computation is syntactic is that the inputs and outputs of a computation often *represent* syntactic entities. We often take an input to a computational process to represent a numeral ('0' or '1') rather than a number (0 or 1). Thus, one often finds the inputs and outputs of an AND gate labelled with the numeral '0' or '1', and this called its 'syntactic content'. Such content may be syntactic, but it is representational content nevertheless.

Another source of confusion about syntax arises from the conflation of the notion of physical computation with the notion of computation in mathematics. As argued in Section 2.2, physical computation and computation in mathematical computation theory are distinct. A Turing machine employs the mathematical notion of computation: it is an abstract mathematical object that does not 'perform' a computation in the same way as a physical system. A Turing machine, typically identified with a set, is not dissimilar in ontological status to a mathematical function, such as  $f(x) = x^2$ . A Turing machine

operates on numerals instead of numbers. It takes numerals (symbols) as input and yields numerals (symbols) as output. A Turing machine operates on syntactic entities.<sup>24</sup>

Two points should be made about these syntactic entities. First, syntactic entities such as numerals are themselves abstract objects—they are not identical to ink-marks on the page, although ink-marks may represent them. Second, syntactic entities are commonly thought of as uninterpreted in this context, that is, as lacking representational content themselves. Hence, mathematical computation does not need to operate on entities that represent. This may lead one to conclude that the computations performed by physical systems do not need to operate on entities that represent either. Unfortunately, this is not true. Although symbolic entities, such as numerals, provide a way of individuating Turing machines, these abstract objects are not available in the physical world. In physical computation, we are stuck with physical stuff, such as ink-marks and electrical impulses. As we have seen, the only way for such stuff to support a plausible notion of computational equivalence is to employ the notion of representation. The non-representational nature of mathematical computation does not carry over to real-world computation. The two notions of computation have different commitments.

## 4 Conclusion

What is the difference between a computation and any other physical process? Under what conditions are two computations the same or different? Partial answers to these questions are now available. A computation essentially involves the manipulation of representations. Computations are consistent ways of mapping representational content to representational content. There is no restriction on the *type* of representational content involved: it may be mathematical, environmental, proximal, distal, broad, narrow, etc. Two physical systems compute the same function—they are I/O equivalent—just in case they map the same representational content to the same representational content. I/O equivalence is a necessary but not a sufficient condition for computational identity. These answers are partial for two reasons. First, not every process that consistently maps representations to representations is a computation. A computation should also, in a sense yet to be defined, be *mechanical*: there should be a counterfactually robust inter-linked series of steps in how it achieves its mapping of representational content. Second, two physical systems perform the same computation just in case they are I/O equivalent *and* they achieve their mapping between representational content *in the same way*. Explicating this latter notion requires considering whether the inter-linked series of steps are appropriately similar in the two cases. There is opportunity here for representational content to enter into the story again. However, a detailed treatment of this condition is the subject matter of a full-blown positive account of computation. For the moment, it should be clear that the representation condition is an essential condition in such an account, and that the arguments against it canvassed above do not hold up.

---

24. See Boolos, Burgess and Jeffrey (2002), pp. 24–25, for more on this point.

## Acknowledgements

I would like to thank all the participants at the Conference on Computation and Cognitive Science for comments on an earlier version of this paper. Particular thanks are due to Ken Aizawa, Frances Egan, Gualtiero Piccinini, Oron Shagrir, and Dan Weiskopf for extended discussion.

## References

- Bechtel, W., and R. C. Richardson. 1993. *Discovering Complexity: Decomposition and Localization as Scientific Research Strategies*. Princeton, NJ: Princeton University Press.
- Block, N. 1990. 'Inverted Earth'. *Philosophical Perspectives* 4:53–79.
- Boolos, G., J. P. Burgess and R. C. Jeffrey. 2002. *Computability and Logic*. 4th ed. Cambridge: Cambridge University Press.
- Burge, T. 1986. 'Individualism and psychology'. *Philosophical Review* 95:3–45.
- Chalmers, D. J. 1996. 'Does a rock implement every finite-state automaton'. *Synthese* 108:309–333.
- Churchland, P. S. 1986. *Neurophilosophy*. Cambridge, MA: MIT Press.
- Clark, A., and D. J. Chalmers. 1998. 'The extended mind'. *Analysis* 58:7–19.
- Crane, T. 2003. *The Mechanical Mind*. 2nd ed. London: Routledge.
- Craver, C. F. 2007. *Explaining the Brain*. Oxford: Oxford University Press.
- Cummins, R. 1989. *Meaning and Mental Representation*. Cambridge, MA: MIT Press.
- Davies, M. 1991. 'Individualism and perceptual content'. *Mind* 100:461–484.
- Dennett, D. C. 1971. 'Intentional Systems'. *The Journal of Philosophy* 68:87–106.
- . 1987. *The Intentional Stance*. Cambridge, MA: MIT Press.
- Egan, F. 1991. 'Must psychology be individualistic?' *Philosophical Review* 100:179–203.
- . 1992. 'Individualism, computation, and perceptual content'. *Mind* 101:443–459.
- . 1994. 'Individualism and vision theory'. *Analysis* 54:258–264.
- . 1995. 'Computation and content'. *Philosophical Review* 104:181–204.
- Fodor, J. A. 1980. 'Methodological solipsism considered as a research strategy in cognitive psychology'. (Reprinted in D. M. Rosenthal, editor, *The Nature of Mind*), *Behavioral and Brain Sciences* 3:63–109.
- . 1987. *Psychosemantics*. Cambridge, MA: MIT Press.
- . 1998. *Concepts*. Oxford: Blackwell.
- Gandy, R. O. 1988. 'The confluence of ideas in 1936'. In *The Universal Turing Machine: A Half-Century Survey*, edited by R. Herken, 55–111. Oxford: Oxford University Press.

- Glennan, S. 2002. 'Rethinking mechanistic explanation'. *Philosophy of Science* 69:S342–S353.
- Kitcher, P. 1988. 'Marr's computational theory of vision'. *Philosophy of Science* 55:1–24.
- Machamer, P. K., L. Darden and C. F. Craver. 2000. 'Thinking about mechanisms'. *Philosophy of Science* 67:1–25.
- Marr, D. 1982. *Vision*. San Francisco, CA: W. H. Freeman.
- Morton, P. 1993. 'Supervenience and computational explanation in vision theory'. *Philosophy of Science* 60:86–99.
- Newell, A. 1982. 'The knowledge level'. *Artificial Intelligence* 18:87–127.
- Piccinini, G. 2008. 'Computation without representation'. *Philosophical Studies* 137:205–241.
- Putnam, H. 1988. *Representation and Reality*. Cambridge, MA: MIT Press.
- Pylyshyn, Z. W. 1984. *Computation and Cognition*. Cambridge, MA: MIT Press.
- Searle, J. R. 1990. 'Is the brain's mind a computer program?' *Scientific American* 262:20–25.
- . 1992. *The Rediscovery of the Mind*. Cambridge, MA: MIT Press.
- Segal, G. 1989. 'On seeing what is not there'. *Philosophical Review* 98:189–214.
- . 1991. 'Defence of a reasonable individualism'. *Mind* 100:485–493.
- . 1997. 'Review of Wilson, R. A., *Cartesian Psychology and Physical Minds*'. *The British Journal for the Philosophy of Science* 48:151–156.
- Shagrir, O. 2001. 'Content, computation and externalism'. *Mind* 110:369–400.
- Shapiro, L. 1997. 'A clearer vision'. *Philosophy of Science* 64:131–153.
- Sieg, W. 1994. 'Mechanical procedures and mathematical experience'. In *Mathematics and Mind*, edited by A. George, 71–117. Oxford: Oxford University Press.
- . 2001. 'Calculation by man and machine: Conceptual analysis'. In *Reflections on the Foundations of Mathematics (Essays in Honor of Solomon Feferman)*, edited by W. Sieg, R. Sommer and C. Talcot, 387–406. Volume 15 of *Lectures Notes in Logic*, Association of Symbolic Logic.
- Sprevak, M. 2007. 'Chinese rooms and program portability'. *The British Journal for the Philosophy of Science* 58:755–776.
- Stich, S. P. 1983. *From Folk Psychology to Cognitive Science*. Cambridge, MA: MIT Press.
- Sudkamp, T. A. 1998. *Languages and Machines*. 2nd ed. Reading, MA: Addison-Wesley.
- Wilson, R. A. 1994. 'Wide computationalism'. *Mind* 103:351–372.